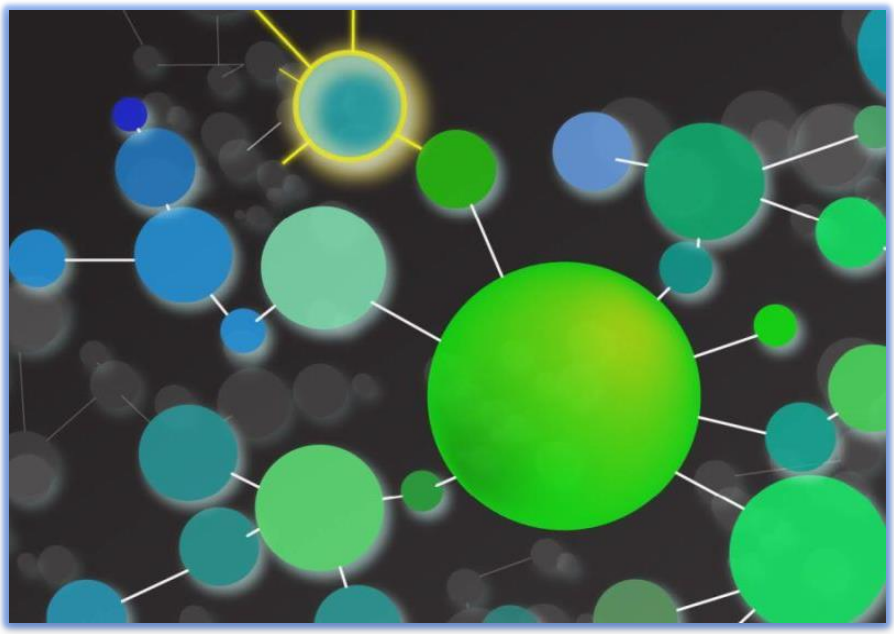
# LIME: Lean Information Management Engine

## 1) Information Model

LIME employs a fractal Information Model meaning that there is no implicit structure, and that all information consists of other information items, including those to govern item features and behaviour. It is thus a graph whose nodes and edges represent Items with Attributes and their Relationships, respectively.

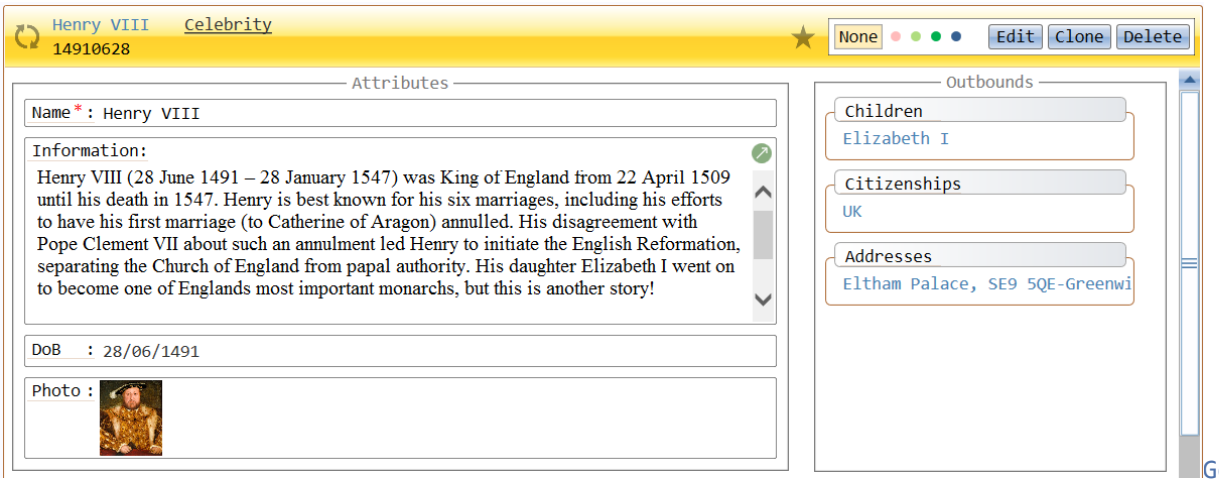


LIME Information Model

The openness of the LIME Information model allows you to implement any Item Structure that serves your purpose. And you do not have to be an expert to use it, either. As long as you know what Properties your Items shall have to start with you can structure your world - without any programming. And you can add and alter Item Definitions as you go along and understand more about your work or have a good idea of what else could be useful to include in the Model. This allows for a controlled evolution of the Model, as all Items are protected by Policies defining the permissions granted to the users defining what Actions they can perform at what stage of the Workflow governing the Item's Lifecycle.

### Items

The Figure below shows the generic view of a Person*. The Information is captured in Properties, which can be Attributes and Relationships. Attributes comprise all the information, which is specific to the Item and no longer relevant once the Item is gone. An example would be the name of a Person, their date of birth or eye colour. Relationships define how 2 Items are connected and why. For example, a person might have Children, be a Citizen of a particular country, and live at one or more Addresses. All these Items are related to that Person, but also do exist independently from it.

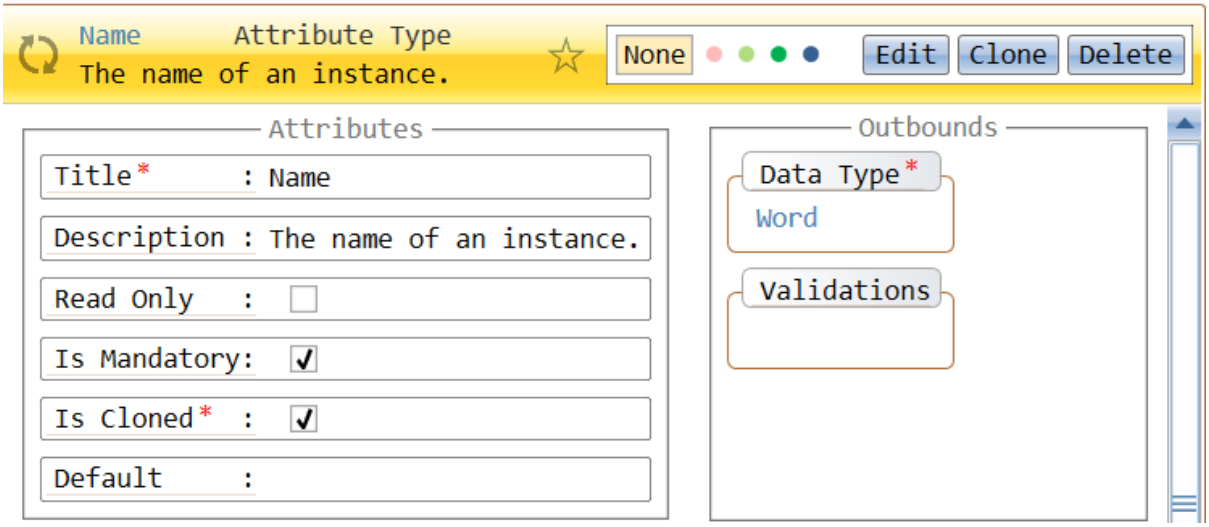



Generic View of an Item

## Attributes

Attributes are the smallest information unit in the LIME Information Model and contain a single value of a particular Data Type (text, number, date, file name, true/false, etc.). There are two types of Attribute Definitions: 'Attribute Types' and 'Value Types', both of which are again Items (don't worry, if this sounds confusing, but this is the fractal nature of the Data Model: every element is an Item, some of which have a special Role in the Model.)

### Attribute Types



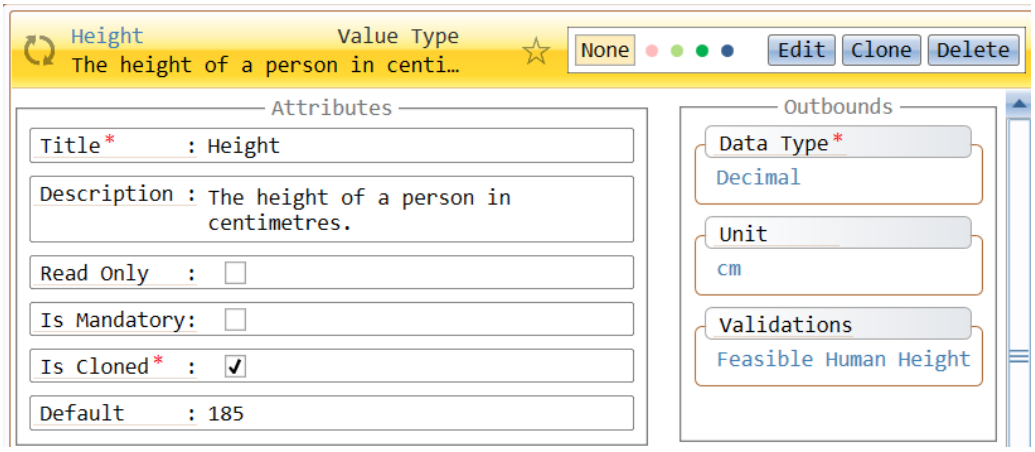

Definition of an Attribute Type

Both define a Title and a Description, as well as other Attributes: *Read Only*, e.g. for automatically assigned IDs, *Mandatory*, and if it is passed on when the containing Item is *Cloned*. You can also set a *Default Value* for newly created Items.

The Properties of an 'Attribute Type" are *Data Type* and *Validation Rule*. While the former is mandatory, the latter is optional and used to prevent wrong data entry by the user. An example would be to limit the allowed dates for the reign of a Soviet leader to a range between 1922 and 1991, the time when the Soviet Union actually existed.

### Value Type

For Attributes that would typically be specified using a unit, like 'meter' or 'second', you can define a Value Type', which is an 'Attribute Type' featuring a unit. All units must be based on an SI-Unit so that they can be used in Calculations:

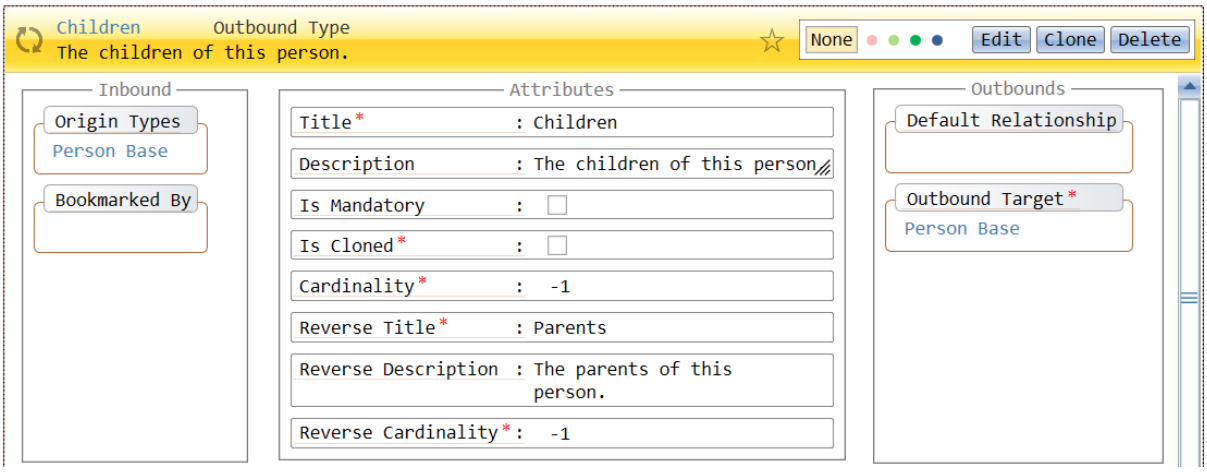



Definition of a Value Type

## Relationships

Most of the information available on a form are references to other items from your information model (shown as links).

In LIME, all relationships are bi-directional and carry different semantics depending on which way you are looking at them. For example, a person can be the parent of another person. The relationship representing this would be called "Children", looking from the parent, and "Parents" looking back from the child. The former is defined as a so-called "Outbound" and the latter as an "Inbound" relationship, though both are in fact the same relationship:




Definition of a Relationship

### Defining a Relationship

The Attributes of a Relationship Type Comprise *Title*, *Description* and *Cardinality* for either Direction of the relation, as well as *Mandatory* and *Cloned*. A Cardinality of –1 means "unlimited".

The bi-directionality of the Relationships is nicely illustrated in the above Figure: on the left, you have the "Inbound" Connections. Here, the Type whose Relationships include this Definition (*Person Base*). On the right are the "Outbound" Relationships, here the Target Type of the Relation, i.e. what type of Items you can connect to. In this example, it is also *Person Base*, as Parent and Child are of the same Type. Finally, there is an option to define a Default Relationship for newly created Items (which makes little sense for "Children").

The following Table is an illustration of 'Parent-Child' relationships. The highlighted Henry VIII famously had two Children who became the first Queens of England, Mary I,

"Bloody Mary" and Elizabeth I, neither of whom had Children of their own. Henry's Parents were Henry VII and Elizabeth of York.

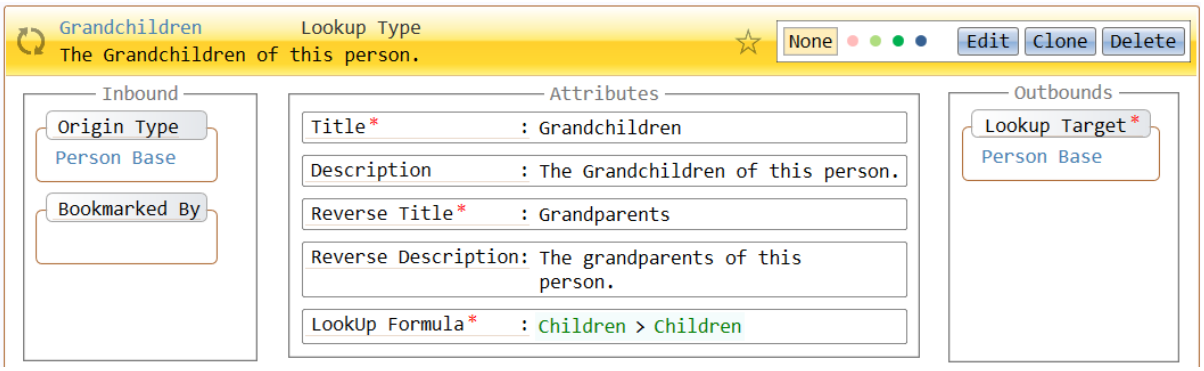
| Item | Image | Name | DoB | † | Children | Parents | Grandchildren | Grandparents |
|---|---|---|---|---|---|---|---|---|
| Mary I |  | Mary I | 18/02/1516 | 17/11/1558 | | Henry VIII | | Henry VII<br>Elizabeth of York |
| Anne Boleyn |  | Anne Boleyn | 01/07/1501 | 19/05/1536 | Elizabeth I | | | |
| Henry VII |  | Henry VII | 28/01/1457 | 21/04/1509 | Henry VIII | | Elizabeth I<br>Mary I | |
| Elizabeth of York |  | Elizabeth of York | 11/02/1466 | 11/02/1503 | Henry VIII | | Elizabeth I<br>Mary I | |
| Henry VIII |  | Henry VIII | 28/06/1491 | 28/01/1547 | Mary I<br>Elizabeth I | Henry VII<br>Elizabeth of York | | |
| Elizabeth I |  | Elizabeth I | 07/09/1533 | 24/03/1603 | | Anne Boleyn<br>Henry VIII | | Henry VII<br>Elizabeth of York |

Item List "Celebrities"

## Lookups and Rearviews

The table also shows indirect relationships that are derived from the defined 'Parent-Child' type, namely 'Grandchildren' and 'Grandparents'. They are the result from defining a computed value called 'Lookup', which is also bi-directional with its reverse values shown as 'Rearviews'.

The 'Grandchildren' Lookup Type is defined in a similar way to a Relationship but has the all-important Attribute 'Lookup Formula', which allows to select the target relationship path (here: 'Children > Children', meaning that the target is found by finding all the children of a person and then all their children, thus yielding all grandchildren. Accordingly, the corresponding Rearview Path is simply the reverse, i.e. 'Parents > Parents', thus yielding all Grandparents of a Person.

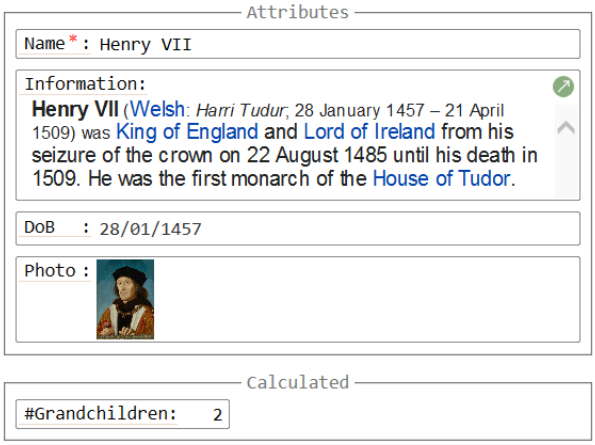


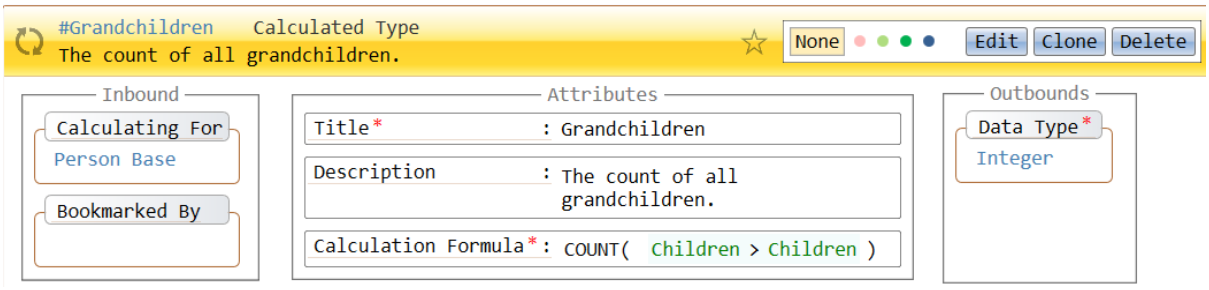

Definition of a Lookup Type

## Calculated Values

LIME allows you to add simple calculations to your Item information. In the example to the right, we calculate the number of Grandchildren for Henry VII, 1father of Henry VIII and Grandfather of Mary I and Elizabeth I.

Defining the Calculation Formula is similar to defining a Lookup. It starts with a Keyword (SUM, AVG, MAX, MIN, and COUNT) followed by the path to the calculation target (see Figure below)



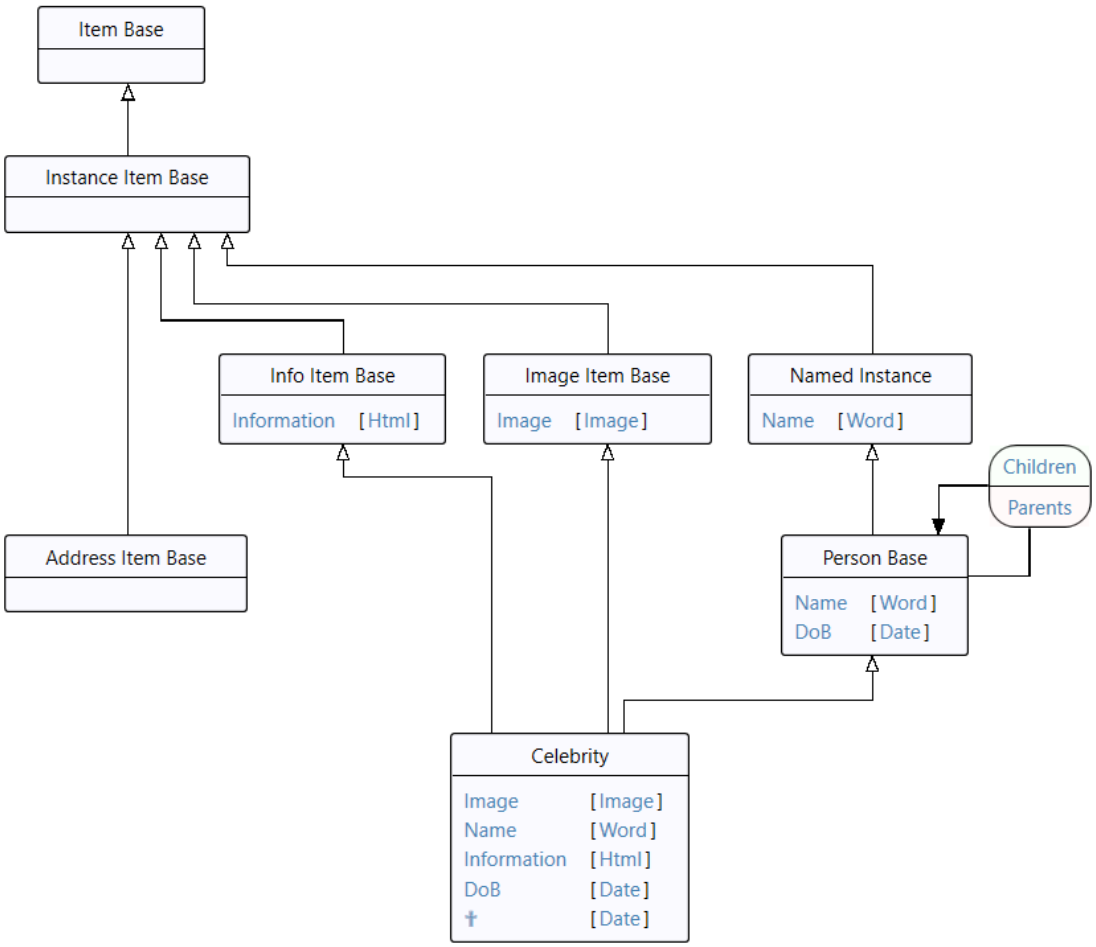



Definition of a Calculation Type

## Inheritance

A good model consists of many small but well managed Items that are connected in a meaningful way to yield a larger, equally well managed set of information. Therefore, an important aspect of information management is reuse. This includes reuse of Instance Information (by pointing to the same Instance when selecting a Relationship thus gaining access to all its information) as well as the reuse of Type defining Properties.

Properties are reused by Types inheriting all the Properties defined on their Parent Types (Multiple Inheritance), as can be seen in the Figure below. Note that the Attributes are shown on all Types, while the Relationships only feature where they are defined, as their repetition would mean considerable cluttering of the diagram.




Information Inheritance Model

## 2) Safety and Security Model

Security and Safety hazards of an Information System relate to control of access to and integrity of all information Items, respectively. In other words, the System must ensure that the Information is protected from unauthorised access and correct. LIME has in-built control mechanisms to provide the highest possible protection from Security and Safety Hazards to minimise the associated risks.

## Access Control

LIME controls access to its information Items using 6 levels of standard permissions to protect information from the eyes of unauthorised users (1-3) and protect Items from misuse:

1) NONE: users[1] cannot find the Item, nor will it appear as a Relationship on any other Item. This level is typically employed to hide certain areas of the model from parts of the team, e.g., when the financial information relating to a Part in a Design shall be hidden from the engineers so that it doesn't influence their technical decision making

2) VIEWERS can find, see, and potentially connect to the Item without being able to read its contents. This level is typically employed to allow the organic growth of the model without revealing sensitive information about a particular Item that is part of it, e.g., to record the decision-making outcome without revealing the discussions leading to it.

3) READERS can view the Item and read its contents. This level is typically used on all non-sensitive Items shared by the team.

4) COMMENTATORS can read the Item's contents and also add comments to it. This is a useful distinction in most collaboration set-ups that intend to share information but limit the group of people who are invited to comment on it. In LIME comments are not Properties of the Item but Items in their own right connecting to it.

5) EDITORS can do all of the above and edit the Item's contents.

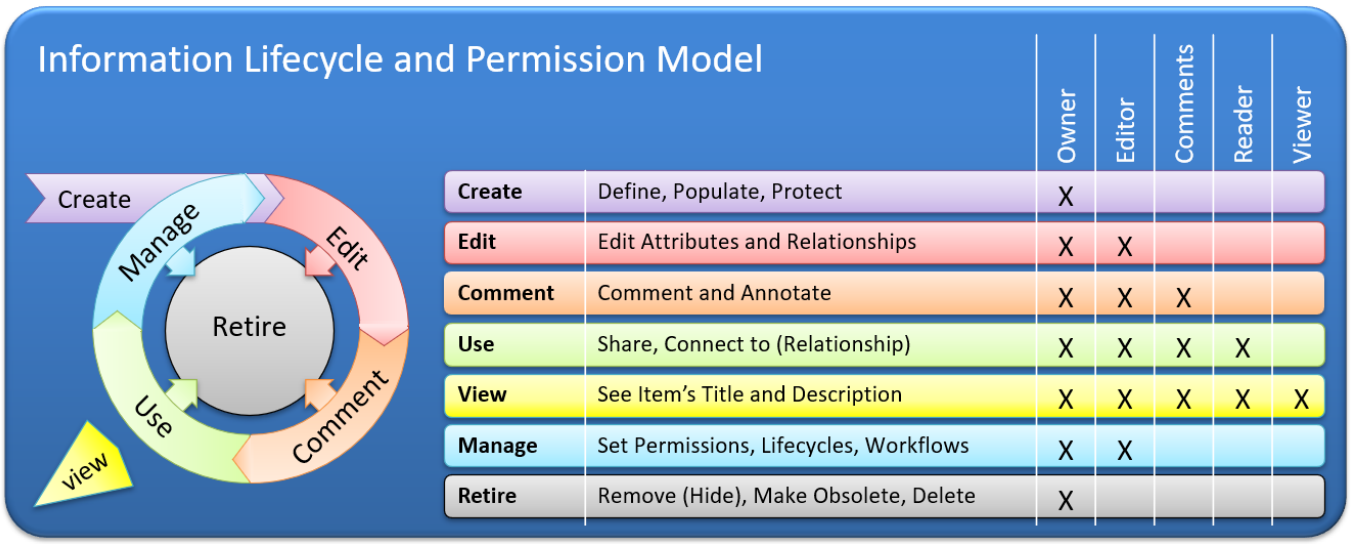



**Fig. 1: Standard Lifecycle and Permission Assignment**

Every Item comes into existence through an act of Creation and Protection (Management), whence its properties are typically edited. Once "live", other users can comment on or connect to it. At the end of its Lifecycle, it will be retired.

[1] Here, a user is synonymous with their account, which identifies their current role, be they human or system user.

6) OWNERS can do all of the above. They are automatically assigned to the Items they create but can hand them over to another user. They are the only ones who can retire Items and change their governance:

```
All Items:  Access and Ownership
Instances:  What Type the Item
Types:      Forms to display or print Instances, Lifecycle and Workflows governing them.
```

# Configuration Control

LIME ensures correctness of its information content through integrity management using mechanisms akin to Configuration Control: It defines the Items' Lifecycles (and sometimes additional Workflows) and the Access Policies mapping the Lifecycle Roles to Users.

## Lifecycles & Workflows

All Items have their Lifecycle, i.e., the various States, in which they can exist. The transition from one State to another is the result of an Event that is in turn the result of an Action executed on the Item. For example, an Item can go from "Draft" to "Editing" when the Action "Edit" is called.

### Lifecycle Modelling

A Lifecycle is in essence a State Machine and can therefore be modelled in the same way. The simplest example comprises the States "Draft" and "Editing" using the Actions "Edit", "Save", and "Cancel" to transition between them:

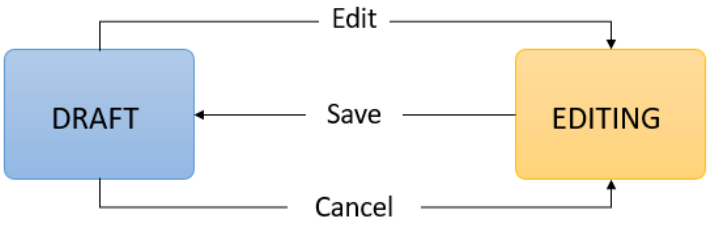



**Fig. 2: Transitioning between the States of a simple Lifecycle**
Like in a State Machine, the Transitions between Lifecycle States have to be clearly defined. Each is triggered by a defined Action.

To control the transitions, it must also be defined, "who" can call these Actions when. As every Item might require a different user to be able to edit it, the "who" has to be defined as an abstract Role, e.g., that of an "Editor" who can execute "Edit" only when the Item is in the "Draft" State, but not when it is in "Editing".

The Lifecycle can thus be modelled as a set of Rules: **Which Role** can Execute **whichever Action** when the Item is in **what State**. Most Roles are static, i.e., they always mean the same - like "Editor". Some Roles, however, can only be calculated based on the current State and previous Actions. A good example for this is the "Current Editor", i.e., the user who is an "Editor" for an Item Lifecycle AND also the one who has executed "Edit" when the Item was in the "Draft" State, so that is now in the "Editing" State.

The above Lifecycle is thus defined by the following two Rules:

1. When an Item is in "Draft", any "Editor" can execute "Edit" to bring it into "Editing".
2. When an Item is in "Editing", only the "Current Editor" can execute "Save" or "Cancel" to bring the Item back into the "Draft" State.

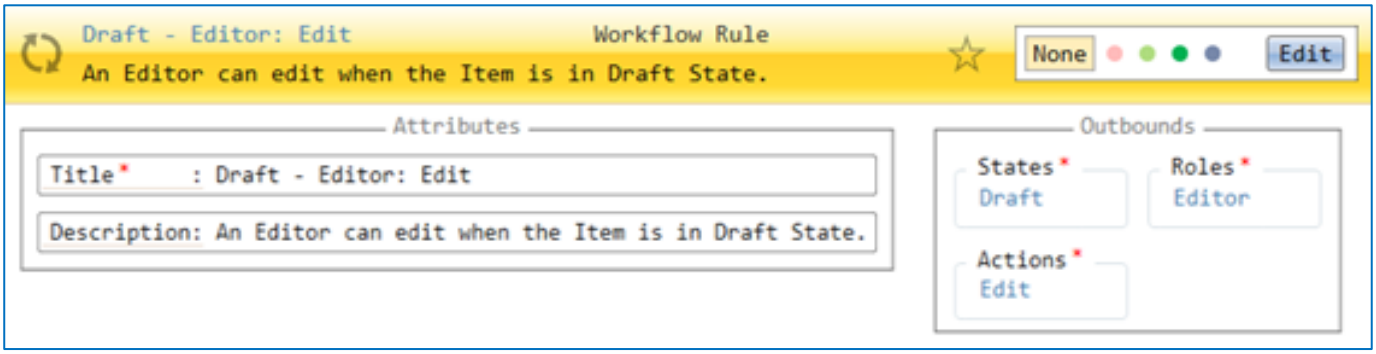



**Fig. 3: Workflow Rule**

A Rule defines, which *Role(s)* can execute whichever *Action(s)* when the Item is in what *State(s)*. To avoid confusion, it is advised to only define a Rule for a single State.

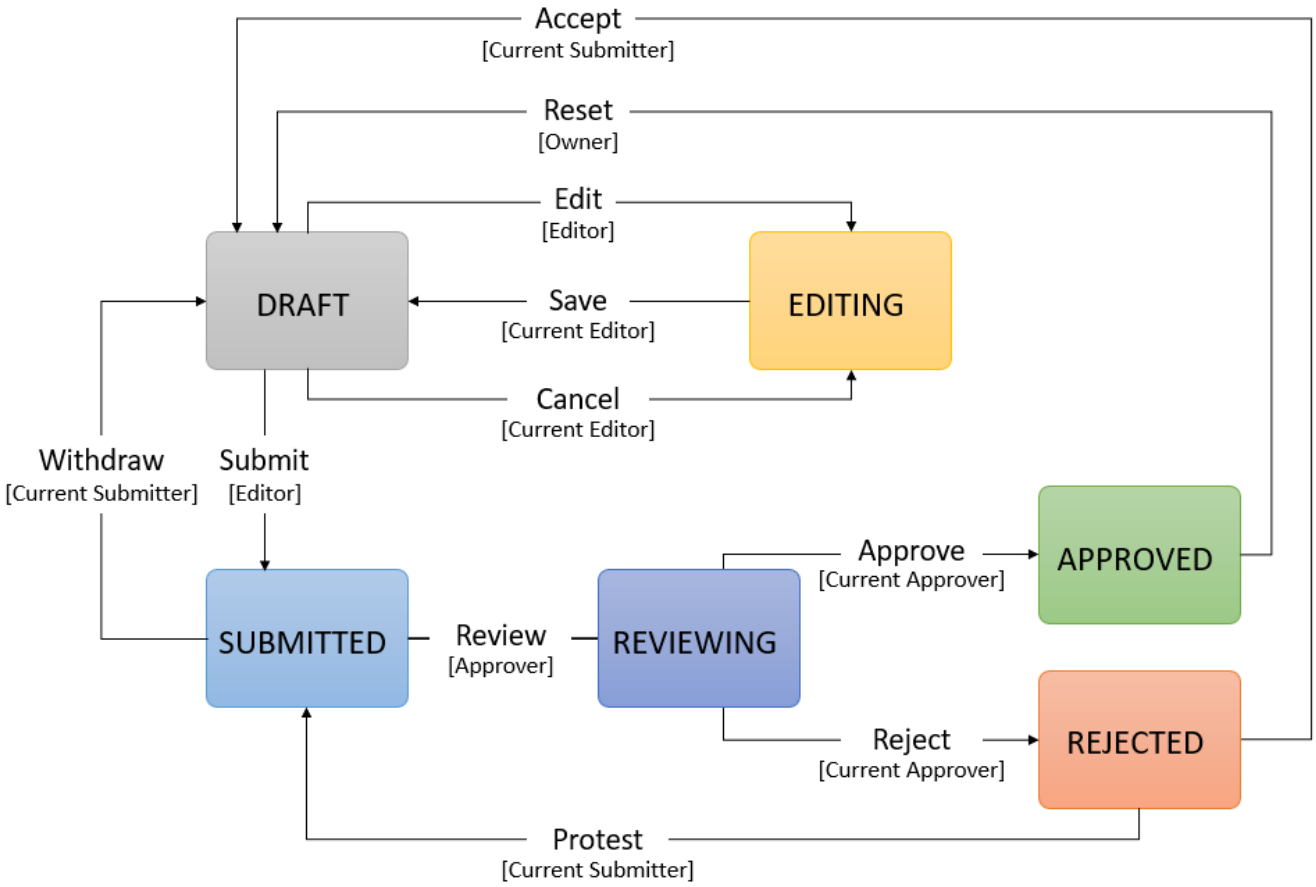



**Fig. 4: Approval Lifecycle**

Whenever an Item enters an 'Active' Lifecycle State, i.e., one during which a user has to perform work on the Item (here: *Editing* and *Reviewing*) assign the user, who put the Item into that State a 'Dynamic' Role (here Current Editor and Current Approver).

If an Item requires a more elaborate Configuration Control, its Lifecycle might be extended to also comprise the States "Submitted", "Approved", and "Rejected" with the additional Actions of "Submit", "Approve", "Reset" and "Reject" to allow moving the Item from one State to the next.

To add this logic to the existing Lifecycle, one only has to add these additional Rules:

- When in "Draft" any "Editor" can execute "Submit" to bring the Item into "Submitted".

- When in "Submitted" only the "Current Submitter" can execute "Withdraw" to bring the Item into the "Draft" State.

- When in "Submitted" any "Approver" can execute "Approve" or "Reject" to bring the Item into the "Approved" and "Rejected" State, respectively.

- When in "Rejected" only the "Current Submitter" can execute "Accept" or "Protest" to bring the Item into the "Draft" and "Submitted" States, respectively.

- When in "Approved" only the "Owner" can execute "Reset" to bring the Item into "Draft".

How one defines the Rules and uses them to define Lifecycles is up to the Configuration objectives for the Items. In LIME, each Item of the same Type will be subject to the same Lifecycle and thus configuration logic.

## Workflow Modelling

Another way of achieving more elaborate Lifecycles is to add Workflows to the Configuration Control. They are ideally orthogonal to the lifecycle, i.e., independent from them. One in-built in LIME is the Registration Workflow for Accounts with a log-in.

The functionality of this Workflow could easily be achieved by adding the Rules to a new Lifecycle. However, as the Registration process is logically orthogonal to editing the name, description etc. of the User account, this is in practice a lot more user friendly. It also allows to map the account to a different person without the need to create an extra account for them.

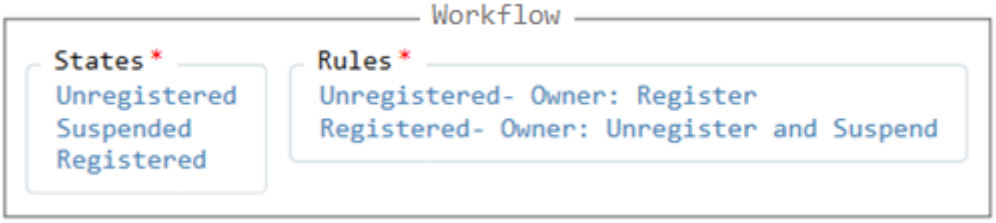



**Fig. 5: The Registration Workflow defined by Rules**
Registering users on LIME does not interfere with their Lifecycle State. Hence, it can be modelled as an independent (orthogonal) Workflow.

## Access Policies

Policies are used to assign Users or Groups of Users to the Roles from Lifecycles and Workflows. A Policy assigns the Users to the standard Roles "Viewer", "Reader", "Commentator", and "Editor" (the "Owner" is not assigned, as it is fundamental to an Item's governance) and to additional Permissions, usually to control Workflow execution. Permissions define what Users or Groups have what Role in which Workflow. This allows for a much more flexible control over the assignment of Roles.

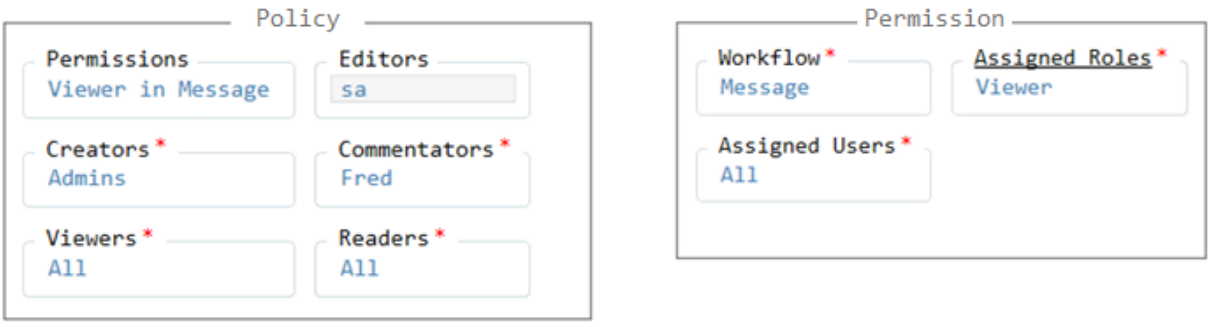



**Fig. 6: Policy and Permission Definitions**

Policies assign Users or Groups to 5 of the 6 Standard Lifecycle Roles (Ownership is not governed by Access Control). It may also comprise any number of additional Permissions assigning Users or Groups to Roles in the context of a defined Workflow.

## Summary

As with all concepts in LIME, everything can be realised through modelling. Even the elements required for defining a Lifecycle or a Workflow, and those governing an Item's access are simply other Items with a particular semantic that they get from their relationships.

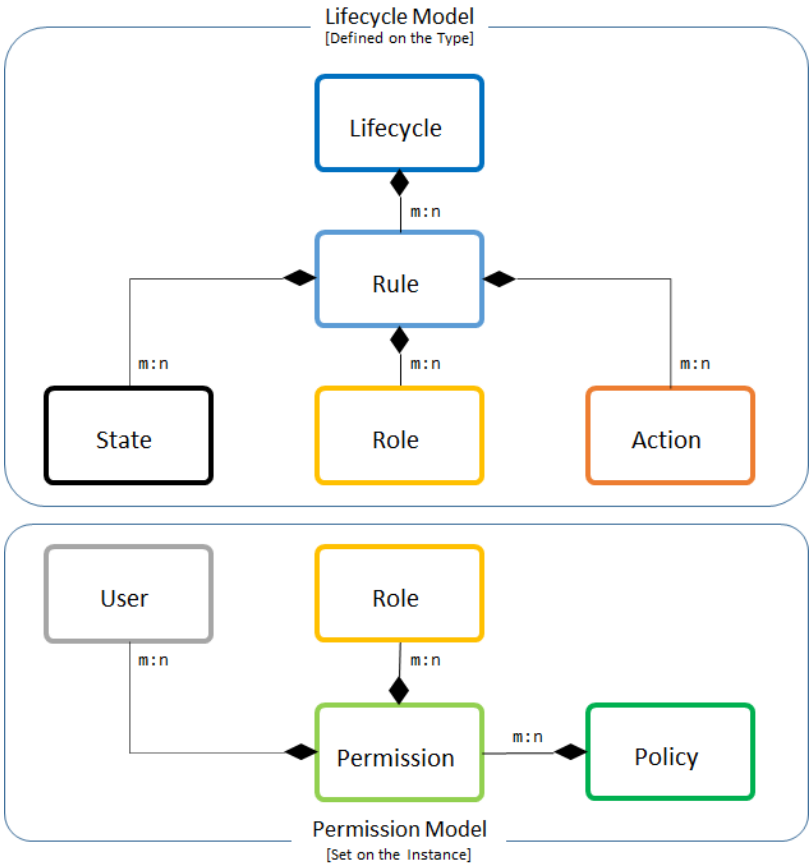



**Fig. 7: Overview of the LIME Safety and Security Model**

A Lifecycle is composed of a set of Rules (m:n), which in turn is a composed of States, Actions and Roles (all m:n) as indicated by the diamond symbol. Permissions assign a User or Group to a Role governing access to the Item.

This makes Safety and Security control highly flexible. One can define custom Workflows and use them for any Item Type. Similarly, one can define and mix the Rules, create Users and Groups to fine-tune whichever Configuration Control Scenario is required to guarantee the maximum protection and integrity of the Information.

It cannot be emphasized enough that - apart from the standard, minimum Lifecycle definitions, every user can define a workflow that suits their purpose without others having to use them (or even know about them, remember the Access Control logic: you can also set the permissions for others to "none").

Every Lifecycle is technically just a Workflow that is an Item composed mainly from Rules, which in turn is an Item composed of States, Actions and Roles.

## Appendix

The following Figure shows the main Attributes and Relationships in the actual Information Model used for Workflows and Lifecycles in LIME. It can be seen, that a Workflow (remember: a Lifecycle is just a Workflow that is used for Lifecycle management but technically the same thing) is basically a set of Rules.

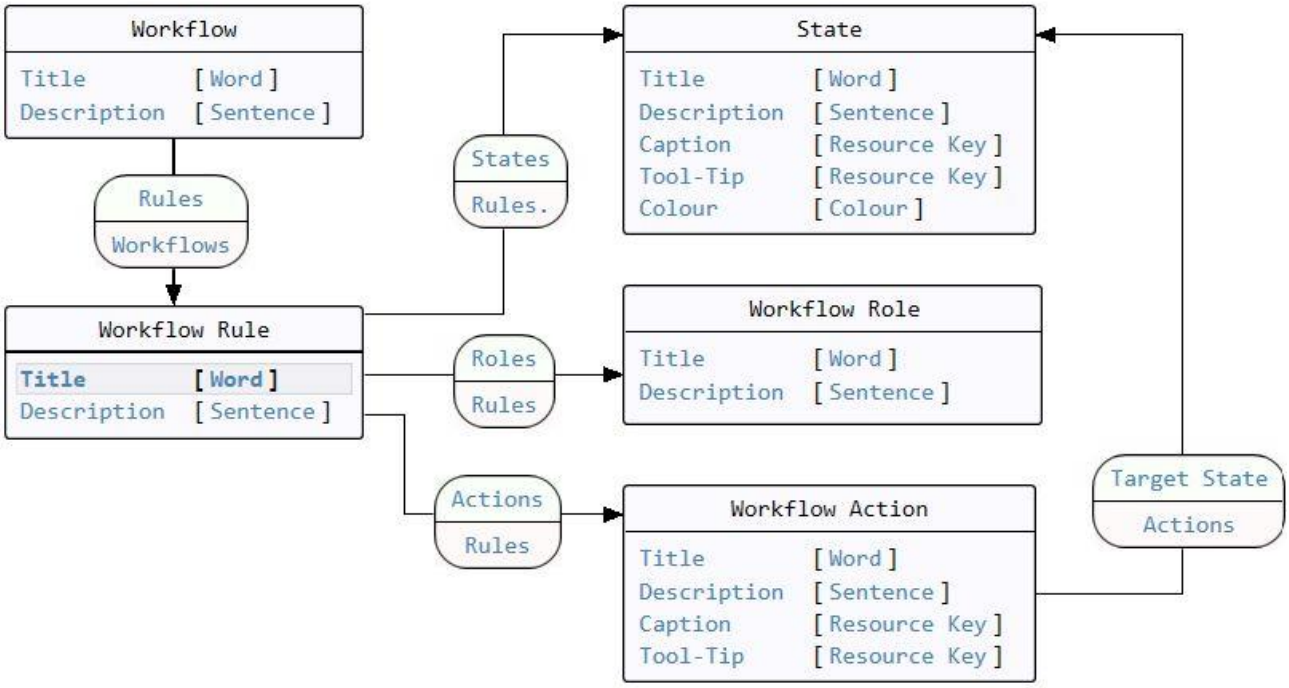



**Fig. 8: The LIME Workflow Entity-Relationship Model**